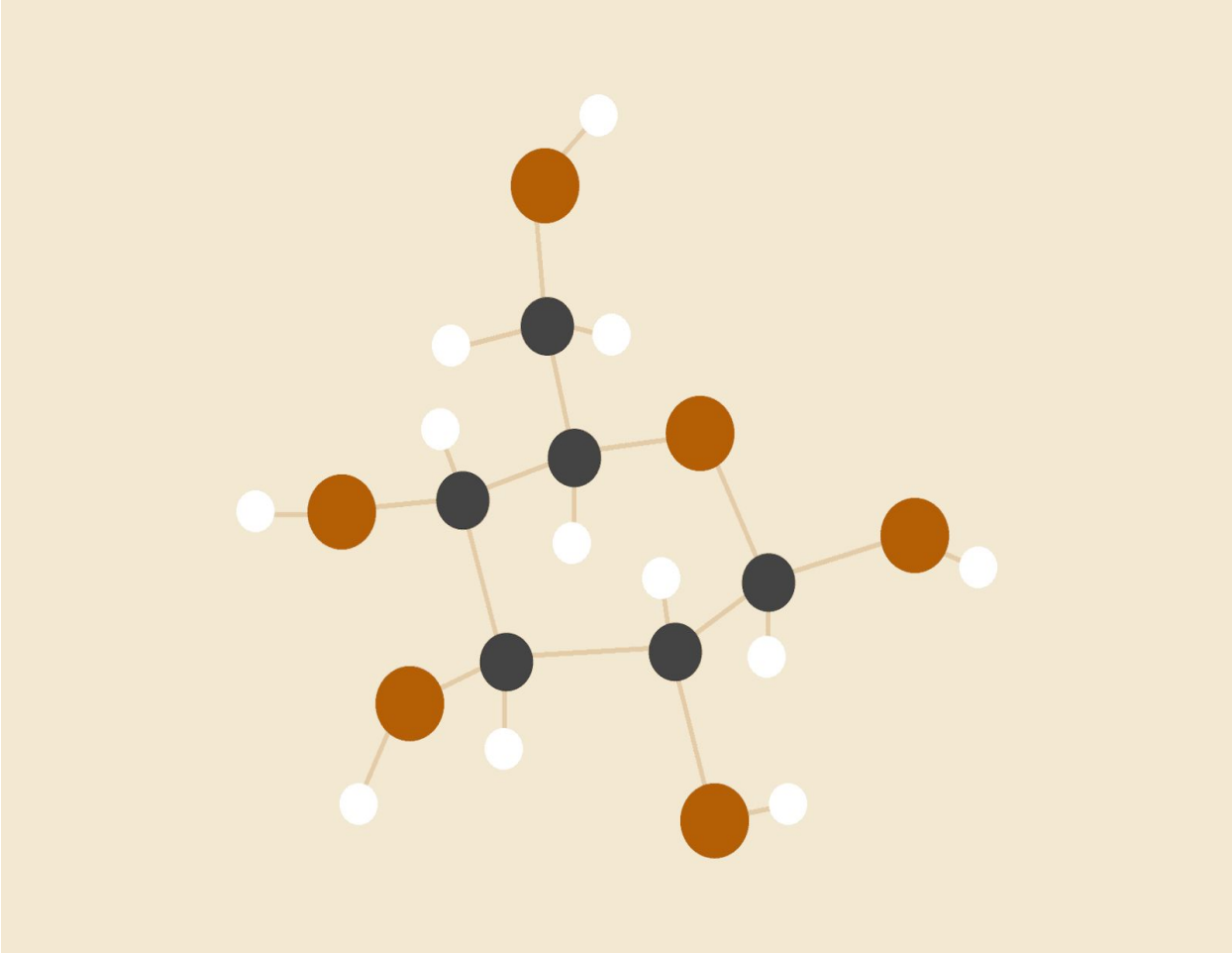
DATA MINING FINAL REPORT

Is Your Product Listing Attractive?



Eashan Adhikarla

12.09.2018
Section 447

INTRODUCTION

To build a model to automatically predict the quality of set of product listing. I used a keras tensorflow model to learn the train data from it's behaviour with the help of the following features;

- name (Name of the product)
- description (Description about the product)
- lvl1, lvl2 and lvl3 (Are the branches of categorization of each product)
- price (Price of the products)
- type (International or a Local product)

PROBLEM STATEMENT

"On E-commerce sites, the quality of product listing is crucial for improving search relevance and gaining customer attention. In this competition, you are provided a set of product names, description, and attributes, as well as the quality score of their listing as rated by real customers."

STAGE 1 - DATA CREATION / PREPARATION

1. Methods, I used for data cleaning includes :

   Tokenizer package from keras's processing text package. Where tokenizer has an attribute "filter" which filters all the special characters in the numpy array. This is why we will also have to first convert the training data from pandas dataframe to numpy array or pandas Series. Keras text processing has a buffer which takes block wise numpy input known as tokens and filters it. Other things I tried for filtering includes :

   Replace technique with regex expression for removing special characters. This glimpse of code is given below;

```
df_test.replace(regex={'<.*?>| |\W':          '          '},
inplace=True)
```

   This method works perfectly fine with both numpy arrays and pandas dataframes.

2. For feature engineering, I started with turning the characters of each word into lower case strings for removing the ambiguity of considering the “Laptop” and “laptop” as different things.
3. I also used feature scaling technique in my file “feature_engineering.ipynb”. Where using min() max() function on data feature “price” helped me a lot to increase my accuracy. I would rate it as an important feature as the accuracy change from drastic form log loss 1.10024 to log loss 0.63879.

```
Cuba Heartbreaker Eau De Parfum Spray 100ml/3....
32GB USB 3.0 Swivel Flash Drive Memory Stick S...
Eican ALP8L-01 Metal + PC Phone Cover Luxury A...
IPAKY Hybrid Luxury Silicone Case Cover And Pl...
Phone case for iPhone 5/5s/SE You Can Be Sore ...
NG-40C Ring-Shaped 40W 3166lm 5400K Macro Phot...
Asus TP300LJ-DW004H Transformer Book Flip 4GB ...
```

STAGE 2 -MODELS

1. Successful models used are as follows:
    a. Feature Engineering and classification with parallel logistic regression with GridCV to search for best scores.To do a linear regression we are trying to do

$$minimize \quad \| X\beta - y \|^2$$

The derivative is ;

$$2X^T (X\beta - y)$$

In small data settings, we can set the derivative to 00 and solve it directly. (e.g., QR decomposition in R.) In big data settings, the data matrix XX is too big to be stored in memory, and may be hard to solve directly. (I am not familiar with how to do QR decomposition or Cholesky decomposition for huge matrices).

One way to parallelize this is by trying to use an iterative method: stochastic gradient descent, where we can approximate the gradient using a subset of the data. (If we use XsXs, ysys to represent a subset of the data, the gradient can be approximated by $2X_s^T (Xs\beta - ys)$, and we can update β with the approximated gradient).

In addition, for the R^2 statistic, we can compute R^2 for all data in parallel

or approximate it by using a subset of the data. The glimpse of this code can be seen in file "**main.ipynb**" which is itself the best example to do it.
Accuracy achieved : log loss 0.57502.

b. Tensorflow model with keras feature embedding of (20000,18).

```
Epoch 11/13
16140/16140 [==============================] - 0s 31us/step - loss: 0.4401 - acc: 0.8062 - val_loss: 0.4857 - val_acc
: 0.7685
Epoch 12/13
16140/16140 [==============================] - 1s 32us/step - loss: 0.4254 - acc: 0.8107 - val_loss: 0.4811 - val_acc
: 0.7675
Epoch 13/13
16140/16140 [==============================] - 1s 33us/step - loss: 0.4109 - acc: 0.8175 - val_loss: 0.4787 - val_acc
: 0.7745
```

The model was a little weird with the dataset but overall it performed better than my other models hence this is my uploaded model. "**Final.ipynb**".

Accuracy achieved : log loss 0.45931.

2. Failed models used are as follows:

   a. Using Gensim 'word2vec', After reading it's documentation I came to realise that using it with keras model is one of the perfect fits for the dataset we had and it would give me the best accuracy possible relatively. But, I failed at implementing it due the the accuracy I achieved was very low due to low epoch number. For increasing the epoch number I needed to run the project on lehigh server and it allowed me 5GB of data which was not enough to have the gensim word2vec pretrained model to be imported with other things to work. I it would take a lot of time to get accuracy in my local machine and after reading some article I came to know training shouldn't be done in local machine but on GPU. But, my model isn't failed it is a working model.
   b. Using Keras text processing with Keyed Vectors was another approach for me to solve the problem. I failed at implementing this model and will try to implement it in future.

3. Possible causes for my success/failure

   a. I had devoted most of my time reading the documentations which actually helped in the long term. Although I was unsuccessful to reflect my research study in this final project.

STAGE 3 - RESULTS

1. The result was a prediction probability which lies between 0 and 1. The precision shows how confident the model was with the products data. The model's confidence was directly proportional to the attractiveness of the data. The more

the features you create in the model the more informative the model becomes more and more the confidence it gains. The glimpse of the data looks like mentioned below;

	id	score
0	18142	0.903027
1	18143	0.867133
2	18144	0.528902
3	18145	0.943663
4	18146	0.910467
5	18147	0.844827

LESSON LEARNED & CONCLUSION

Thinking more about it, it seems like log loss actually makes sense in the real world. We would want to trust the model and ideally model should admit that it cannot figure out instead of being confidently wrong. I learned how to understand the mathematics behind the model I am using not just by the project but by the CSE447 class I had taken. The class really helped me a lot in broad perspective to think about the background of the problem statement and to implement the best strategy to achieve it.

I thank both Prof. Wang Ting and teaching assistant Yujie Ji for their support and help for successfully completing the course and understanding the complex doubts I had.

I will take this learning for the rest of my research and real life and short term goal is to implement the unsuccessful models for this final project.

REFERENCES

1. https://www.pydoc.io/pypi/gensim-3.2.0/autoapi/models/word2vec/index.html
2. https://github.com/keras-team/keras-preprocessing
3. https://medium.com/@nokkk/how-to-win-kaggles-competitions-data-cleansing-and-features-engineering-part-1-45404c261637
4. https://www.kaggle.com/mihaskalic/lstm-is-all-you-need-well-maybe-embeddings-also
5. https://www.kaggle.com/marijakekic/cnn-in-keras-with-pretrained-word2vec-weights

THANK YOU!